Logic Programming and Reasoning with Incomplete Information

Michael Gelfond
Computer Science Department
University of Texas at El Paso
El Paso, Texas 79968
mgelfond@cs.ep.utexas.edu

January 21, 1994

Abstract

The purpose of this paper is to expand the syntax and semantics of logic programs and disjunctive databases to allow for the correct representation of incomplete information in the presence of multiple extensions. The language of logic programs with classical negation, epistemic disjunction, and negation by failure is further expanded by new modal operators K and M (where for the set of rules T and formula F, KF stands for "F is known to be true by a reasoner with a set of premises T" and MF means " F may be believed to be true" by the same reasoner). Sets of rules in the extended language will be called epistemic specifications. We will define the semantics of epistemic specifications (which expands the semantics of disjunctive databases from [GL91]) and demonstrate their applicability to formalization of various forms of commonsense reasoning. In particular, we suggest a new formalization of the closed world assumption which seems to better correspond to the assumption's intuitive meaning.

VISIT…

1 Introduction

As was demonstrated in recent years, traditional logic programming language provides a powerful tool for knowledge representation. Its main non-monotonic feature - negation as failure [Cla78] - makes it possible to express many interesting types of commonsense knowledge which are not readily expressible in classical logic. Unlike classical logic, however, traditional logic programming does not allow a programmer to directly represent incomplete knowledge about the world. A consistent classical theory partitions the set of sentences into three parts: those which are provable, those which are refutable, and those which are undecidable. A logic program partitions the set of ground queries into only *two* parts: A query is answered either *yes* or *no*. This happens because the syntax of logic programming does not allow the representation of disjunctive information and because the traditional declarative semantics of logic programming automatically applies the *closed world assumption* [Rei78] to all predicates (i.e. each ground atom that does not follow from the facts included in the program is assumed to be false). Procedurally, the query evaluation methods of logic programming give the answer *no* to every query that does not succeed; they provide no counterpart for undecidable sentences, which represent the incompleteness of information in classical axiomatic theories.

The first attempt to lift the syntactic limitation described above is probably due to Jack Minker. In [Min82] he considers positive disjunctive databases defined as collections of rules of the form

$$A_1 \; or \; \ldots \; or \, A_n \leftarrow B_1, ..., B_m \tag{1}$$

where $A's$ and $B's$ are atoms. The type of incompleteness expressible in these databases is, however, rather limited since their semantics suggested in [Min82] implicitly assumes a form of closed world assumption. This work was generalized and/or modified by various authors (an overview can be found in [PP90], [LMR92]) but most of the approaches still assume the closed world assumption and hence does not allow the representation of such simple forms of incompleteness as missing informations in the database tables, null values, partial definitions, etc.

The problem of lifting the closed world restriction of logic programming

was recently addressed in [GL90] [1] where the authors consider "extended" logic programs, that contain *classical (or strong) negation* $\neg$ in addition to negation as failure *not*. General (classical) logic programs provide negative information implicitly, through closed-world reasoning; an extended program can include explicit negative information, as well as explicit closed world assumptions for some of its predicates. In the language of extended programs, one can distinguish between a query which fails in the sense that it *does not succeed* and a query which fails in the stronger sense that its *negation succeeds*. The semantics of extended logic programs is based on the notion of answer sets. This semantics views the program rules as constraints used by a reasoner associated with the program to build possible theories about the world. These theories (called answer sets, or belief sets) consist of literals and therefore are vivid in the sense of H. Levesque [Lev86]. For extended programs without classical negation their answer sets coincide with stable models from [GL88].

In [GL90] we consider primarily well-defined extended logic programs, i.e. extended programs with unique consistent answer sets. The answer such a program returns to a ground query Q [2] is *yes*, *no*, or *unknown*, depending on whether the answer set contains Q, $\neg Q$, or neither. The existence of several answer sets indicates that the corresponding program P has several possible interpretations, i.e. it is possible for a rational reasoner to construct several theories satisfying P. Such a multiplicity becomes a norm rather than exception if the notion of extended logic program and its answer set semantics is expanded to that of extended disjunctive database [GL91] (see also [Prz90]) — collections of rules of the form:

$$A_1 \; or \; \ldots \; or A_n \leftarrow B_1, ..., B_m, not \; C_1, \ldots, not \; C_k \qquad (2)$$

where A's, B's, and C's are atoms p or their "classical" negations $\neg p$. We will assume that rules with variables are used as shorthands for the sets of all their ground instantiations. (Notice the use of symbol *or* instead of classical $\vee$. The meaning of a connective *or*, called epistemic disjunction, is given by the semantics of disjunctive databases and differ from that of $\vee$. The meaning of a formula $A \vee B$ is "A is true or B is true" while a rule

[1] A similar approach was independently developed and investigated in [PW89]. See also [KS90].

[2] Here Q is a literal. In the next section we will consider more complicated queries

$A \; or \; B \leftarrow$ is interpreted epistemically and means "A is believed to be true or B is believed to be true.") Let Δ be the collection of answer sets of a disjunctive database Π. We will say that the *answer* to a query Q is *yes* if every answer set from Δ contains Q, *no* if every answer set from Δ contains the complement of Q, and *unknown* otherwise. (The last answer can be split into several more informative answers but above alternatives are sufficient for the purpose of this paper.)

In [GL90] we argue that for well-defined programs the presence of two types of negation allows one to deal in a natural and convenient way with incomplete information. This, however, is no longer the case if the corresponding programs are not well-defined. **The purpose of this paper is to expand the notions of extended logic programs and disjunctive databases to allow for the correct representation of incomplete information in the presence of multiple belief sets.**

We will start by demonstrating the problem using a modification of the following example from [GL90]:

Example 1

Consider a collection of rules

1. $Eligible(x) \leftarrow HighGPA(x)$,
2. $Eligible(x) \leftarrow Minority(x), FairGPA(x)$,
3. $\neg Eligible(x) \leftarrow \neg FairGPA(x), \neg HighGPA(x)$,
4. $Interview(x) \leftarrow \quad not \; Eligible(x),$
   $not \; \neg Eligible(x)$

used by a certain college for awarding scholarships to its students. The first three rules are self explanatory (we assume that variable x ranges over a given set of students) while the fourth rule can be viewed as a formalization of the statement:

(*) "The students whose eligibility is not determined by the first three rules should be interviewed by the scholarship committee."

In its epistemic form the rule says : $Interview(x)$ if neither $Eligible(x)$ nor $\neg Eligible(x)$ is known. We assume that this program is to be used in conjunction with a database DB consisting of literals specifying values of the

predicates *Minority*, *HighGPA*, *FairGPA*. Consider, for instance, DB consisting of the following two facts about one of the students:

5. $FairGPA(ann) \leftarrow$
6. $\neg HighGPA(ann) \leftarrow$

(Notice that DB contains no information about the minority status of Ann.) Intuitively it is easy to see that rules (1)–(6) allow us to conclude neither $Eligible(ann)$ nor $\neg Eligible(ann)$, therefore eligibility of Ann for the scholarship is undetermined and, by rule (4), she must be interviewed. Formally this argument is reflected by the fact that program T_1 consisting of rules (1)–(6) has exactly one answer set:

$$\{FairGPA(ann), \neg HighGPA(ann), Interview(ann)\}.$$

The situation changes significantly if disjunctive information about students is allowed to be represented in the database. Suppose, for instance, that we need to augment rules (1)–(3) by the following information:

(**) Mike's GPA is fair or high.

The corresponding extended disjunctive database T_2 consists of rules (1)–(3) augmented by the disjunction

7. $FairGPA(mike)\ or\ HighGPA(mike) \leftarrow$

T_2 has two answer sets:

$$A_1 = \{HighGPA(mike), Eligible(mike)\}$$

and

$$A_2 = \{FairGPA(mike)\},$$

and therefore the reasoner modeled by T_2 does not have enough information to establish Mike's eligibility for the scholarship (i.e. answer to $Eligible(mike)$ is *unknown*). If we now expand this theory by (*) we expect the new theory T_3 to be able to answer *yes* to a query $Interview(Mike)$. It is easy to see however that if (*) is represented by (4) this goal is not achieved. The resulting theory T_3 consisting of (1)–(4) and (7) has two answer sets

$$A_3 = \{HighGPA(mike), Eligible(mike)\}$$

$$A_4 = \{FairGPA(mike), Interview(mike)\}$$

and therefore the answer to query $Interview(mike)$ is *unknown*. The reason of course is that (4) is too weak to represent (*). The informal argument we are trying to capture goes something like this: theory T_3 answers neither *yes* nor *no* to the query $Eligible(mike)$. Therefore, the answer to this question is undetermined, and, by (*), Mike should be interviewed. To formalize this argument our system should have a more powerful introspective ability (termed strong introspection in [Gel91]) than the one captured by the notion of answer sets from [GL91]. Roughly speaking instead of looking at only one possible set of beliefs sanctioned by T it should be able to look at all such sets.

Remark. The situation will not change if (**) is represented by modeling disjunctions in the language of logic programs. For instance, replacing (7) by two rules

$$FairGPA(Mike) \leftarrow not\ HighGPA(Mike)$$

$$HighGPA(Mike) \leftarrow not\ FairGPA(Mike)$$

will not change answer sets of the program.

In this paper we extend the syntax of disjunctive databases from [GL91] in two directions. Firstly, following [LT84], and [Wag91] we allow the rules to contain other types of formulae in addition to literals. Secondly, and more importantly, we expand the language by a modal operators K and M. Sets of rules in the extended language are called *epistemic specifications*. We will define the semantics of epistemic specifications (which expands the semantics of disjunctive databases from [GL91]) and demonstrate their applicability to formalization of various forms of commonsense reasoning. The notion of epistemic specification and some of the other material in this paper was first presented in [Gel91]. Our definitions are an improvement over those in [Gel91].

2 Definitions

Let us consider a language $\mathcal{L}_0$ consisting of predicate symbols $p, q, \ldots$, object variables, function symbols, connectives &, $\neg$, $\exists$, and the modal operators K and M where KF stands for "F is known to be true", and MF stands for "F

may be believed to be true." Terms and formulae of $\mathcal{L}_0$ will be defined in the usual way. Formulae of the form $p(t_1, \ldots, t_n)$ where $t_1, \ldots, t_n$ are terms are called atoms. By literals we will mean atoms $p(t_1, \ldots, t_n)$ and their strong negations $\neg p(t_1, \ldots, t_n)$. Literals not containing variables are called ground. The set of all ground atoms will be denoted by *Atoms* and the set of all ground literals will be denoted by *Lit*.

Let us consider a collection $A = \{A_i\}$ of sets of ground literals and a set W of such literals. (A can be thought of as a collection of possible belief sets of a reasoner while W represents his current (working) set of beliefs.) We will inductively define the notion of truth ($\models$) and falsity ($=|$) of formulae of $\mathcal{L}_0$ w.r.t. a pair $M = < A, W >$.

$M \models p(t_1, \ldots, t_n)$ iff $p(t_1, \ldots, t_n) \in W$.

$M \models \mathrm{K}F$ iff $< A, A_k > \models F$ for every A_k from A

$M \models \mathrm{M}F$ iff $< A, A_k > \models F$ for some A_k from A

$M \models F \& G$ iff $M \models F$ and $M \models G$

$M \models \exists x F$ iff there is a ground term t such that $M \models F(t)$

$M \models \neg F$ iff $M =| F$

$M =| p(t_1, \ldots, t_n)$ iff $\neg p(t_1, \ldots, t_n) \in W$

$M =| \mathrm{K}F$ iff $M \not\models \mathrm{K}F$

$M =| \mathrm{M}F$ iff $M \not\models \mathrm{M}F$

$M =| F \& G$ iff $M =| F$ or $M =| G$

$M =| \exists x F$ iff for every ground term t, $M =| F(t)$

$M =| \neg F$ iff $M \models F$

In our further discussion we will expand language $\mathcal{L}_0$ by the connectives *or* and $\forall$ defined as follows:

$(F \; or \; G)$ iff $\neg(\neg F \,\&\, \neg G)$

$\forall x F$ iff $\neg \exists x \neg F$

Formulae of the expanded language $\mathcal{L}$ not containing modal operators will be called *objective formulae*. Formulae constructed from KF and MF (where F is objective) and from logical connectives and quantifiers will be called *subjective*.

It is easy to see that according to the definition above, the truth of subjective sentences does not depend on W while the truth of objective ones does not depend on A, i.e. we have a notion of objective formula being true (false) in W and subjective formula being true (false) in A. We will denote the former by $W \models F$ ($W =\mid F$) and later by $A \models F$ ($A =\mid F$).

The language and the satisfiability relation described above together with the notion of a rule from logic programming will be used to provide a specification of a reasoner with the desired beliefs. (This view on the role of logic in nonmonotonic reasoning seems to be similar to the one advocated by H. Levesque in [Lev90]). Formally, by an *epistemic specification* we will mean a collection of rules of the form

$$F \leftarrow G_1, \ldots, G_m, not\ G_{m+1}, \ldots, not\ G_k \tag{3}$$

where F and $G_{m+1} \ldots G_k$ are objective and $G_1 \ldots G_m$ are subjective or objective formulae.

Now we will define a collection A of sets of ground literals satisfying an epistemic specification T. We will call such a collection a *world view* of T and its elements belief sets of T. The precise definition of these notions will be given in several steps:

Step 1. Let us first assume that T is an epistemic specification **not containing modal operators and negation as failure**. A set W of ground literals is called a *belief set* of such a specification iff W is a minimal set satisfying the following two conditions:

1. For every rule $F \leftarrow G_1, \ldots, G_m$ from T such that $W \models G_1 \,\&\, \ldots \,\&\, G_m$ we have $W \models F$.

2. if W contains a pair of complementary literals then $W = Lit$. (This belief set will be called inconsistent.)

Example 2

Let T consist of the rules:

1. $p(a)\ or\ p(b) \leftarrow$
2. $\neg p(b) \leftarrow$
3. $\exists x\ q(x) \leftarrow$

It is easy to see that T has two belief sets: $\{p(a), \neg p(b),\ q(a)\}$ $\{p(a), \neg p(b),\ q(b)\}$

Step 2. Now let us assume that T is an epistemic specification **not containing modal operators** and let W be a set of literals in the language of T. By T_W we will denote the result of

1. removing from T all the rules containing formulas of the form $not\ G$ such that $W \models G$
2. removing from the rules in T all other occurrences of formulas of the form $not\ G$.

Obviously, T_W contains neither modal operators nor negation by failure and therefore its belief sets are defined in step one. We will say that W is a **belief set** of T if W is a belief set of T_W.

Example 3

Let T consist of the rules

1. $P(a)\ or\ P(b) \leftarrow \exists x Q(x), not\ Q(d)$
2. $Q(c) \leftarrow$

It is easy to see that this specification has two belief sets $\{Q(c),\ P(a)\}$ and $\{Q(c), P(b)\}$.

Step 3. Finally, let T be an **arbitrary** epistemic specification, and **A** be a collection of sets of literals in its language. By T_A we will denote the epistemic specification obtained from T by:

1. removing from T all rules containing formulas of the form G such that G is subjective and $\mathbf{A} \not\models G$,

2. removing from rules in T all other occurrences of subjective formulas.

Definition 1. A set $\mathbf{A}$ will be called a **world view** of T if $\mathbf{A}$ is the collection of all belief sets of T_A. Elements of $\mathbf{A}$ will be called *belief sets* of T. The specification T_A will be called the *reduct* of T w.r.t. $\mathbf{A}$.

Obviously, if T does not contain modal operators then it has a unique world view consisting of all belief sets of T.

Example 4

Let T consist of the rules

1. $Pa \; or \; Pb \leftarrow$

2. $Pc \leftarrow$

3. $Qd \leftarrow$

4. $\neg Px \leftarrow \neg \mathrm{M}Px$

The specification T has three world views:

$A_1 = \{\{Qd,\ Pc,\ Pa,\ \neg Pb,\ \neg Pd\}\}$

$A_2 = \{\{Qd,\ Pc,\ Pb,\ \neg Pa,\ \neg Pd\}\}$

$A_3 = \{\ \{Qd,\ Pa,\ Pc,\ \neg Pd\},\ \{Qd,\ Pb,\ Pc,\ \neg Pd\}\}$

Example 5

Let T consist of the rules

1. $Pa \leftarrow$

2. $Qb \; or \; Qc \leftarrow$

3. $Rx \leftarrow \neg \mathrm{K}Qx$

4. $Sx \leftarrow \neg \mathrm{M}Qx$

The only world view of T is

$A = \{\{Pa, Qb, Ra, Rb, Rc, Sa\},\ \{Pa, Qc, Ra, Rb, Rc, Sa\}\}$

Example 6

Let $T = \{p \leftarrow \neg \mathrm{K}p\}$. It is easy to see that T does not have a world view.

Example 7

Let $T = \{p \leftarrow \neg \mathrm{M}q,\ q \leftarrow \neg \mathrm{M}p\}$. This specification is satisfied by two world views: $A_1 = \{\{q\}\}$ and $A_2 = \{\{p\}\}$.

Definition 2

We will say that a world view of epistemic specification T is **consistent** if it does not contain a belief set consisting of all literals.

Definition 3

We will say that an epistemic specification is **consistent** if it has at least one consistent non–empty world view.

Example 8

Let T consist of rules $p \leftarrow$ and $\neg p \leftarrow$. It is easy to see that specification T is inconsistent. Another inconsistent specification is given in Example 6.

Definition 4

Let T be an epistemic specification and $A = \{A_i\}$ be a world view of T. A formula F is **true** in A ($A \models F$) iff $< A, A_i > \models F$ for every A_i from A.

Definition 5

Let T be an epistemic specification. A formula F is **true** in T ($T \models F$) iff $A \models F$ for every world view A of T.

This definition can be used to define the range of possible answers to a query Q (where Q is an arbitrary formulae of $\mathcal{L}$ without free variables). For the purpose of this paper we will limit ourselves to the simple case when answer to Q is *yes* if $T \models Q$, *no* if $T \models \neg Q$, and *unknown* otherwise.

The following simple proposition establishes the relationship between disjunctive databases and epistemic specifications.

Proposition 1. Let T be an epistemic specification consisting of rules of the form $F \leftarrow G_1, \ldots, G_m, not\ E_{m+1}, \ldots, not\ E_k$.

Then

1. If F, G's and E's are atoms (i.e. T is a general logic program) then A is a world view of T iff A is the set of all stable models of T.

2. If G's, and E's are objective literals and F is a disjunction of objective literals (i.e. T is an extended disjunctive database) then A is a world view of T iff A is the set of all answer sets of T.

Proof. Follows immediately from the definition of world view of T and the fact that for any collection of sets of literals A and any general logic program (or an extended disjunctive database) T, $T_A = T$.

3 Applications

In this section we will discuss several applications of epistemic specifications to formalization of commonsense reasoning. The emphasis will be on the expressive power of the language and not on the computational mechanisms necessary to design efficient query answering systems. Such mechanisms will be discussed elsewhere.

3.1 Representing the Unknown

We will start with demonstrating how statements of the form "**unknown** p" can be represented by strongly introspective formulae. We suggest to represent formulae of this form as a conjunction of statements $\neg Kp$ and $\neg K\neg p$. Let us go back to Example 1 from the introduction to illustrate this point.

Example 1 revisited. Let us consider the theory consisting of rules (1)– (3) and (7) from Example 1. To obtain the proper formalization of the statement

(*) "The students whose eligibility is not determined by the first three rules should be interviewed by the scholarship committee"

we will just replace rule (4) by

4'. $Interview(x) \leftarrow \neg$ K $Eligible(x)$,
$\neg$ K $\neg Eligible(x)$

which corresponds closely to the intuitive meaning of (*). It is easy to check that the theory T consisting of rules (1)–(3), 4', and (7) has the world view $A = \{A_1, A_2\}$ where

$$A_1 = \{HighGPA(Mike), Eligible(Mike), Interview(Mike)\}$$

$$A_2 = \{FairGPA(Mike), Interview(Mike)\}$$

Therefore T answers *unknown* to the query $Eligible(Mike)$ and *yes* to the query $Interview(Mike)$ which is the intended behavior of the system.

3.2 Closed World Assumption

Now we will illustrate how epistemic specifications can be used to formalize the closed world assumption of [Rei78] in the presence of disjunctive information. This question has been extensively studied in the context of various nonmonotonic formalisms. [Min82] gives perhaps the most widely known form of the closed world assumption for positive disjunctive databases called the generalized closed world assumption (GCWA). By now there are many useful generalizations of this assumption expanding the original idea. Most of them tend to interpret disjunction as exclusive. For instance, a positive disjunctive database $\{Pa \ \ or \ \ Pb \leftarrow\}$ will answer "No" to a query Pa & Pb [3]. [RT88] noticed that in some applications disjunctive databases with the semantics based on GCWA or its extensions may lead to unintuitive conclusions, and attempted to remedy the problem by weakening the corresponding assumptions. Some further work in this direction can be found in [Cha89, C.89], etc. In all these approaches however closed world assumption remains a part of the semantics. We argue that to make disjunctive databases a viable language for representing knowledge in the presence of incomplete information the assumption should be removed from the semantics and made expressible as a statement of the language. This is especially clear when our language contains classical (strong) negation. Syntactically,

[3] We follow the database tradition in using the term exclusive here. The term is somewhat misleading since a theory $\{Pa \ \ or \ \ Pb \leftarrow, \ \ Pa \leftarrow, \ \ Pb \leftarrow\}$, which would be inconsistent if *or* were truly exclusive, is consistent in GCWA and other semantics.

general disjunctive databases not containing $\neg$ are a special case of epistemic specification. Moreover, their "canonical" models are identical to their belief sets. In spite of this, there is a semantic difference between a set of rules viewed as a general database, and the same set of rules viewed as an epistemic specification. The absence of a ground atom Q in a "canonical" model of a general database indicates that Q is false, so that the correct answer to the query Q is *no*; the absence of Q in the corresponding answer set of the same collection of rules treated as an extended program indicates that the answer to this query should be *unknown*.

Example 9 The epistemic specification Π

$Pa \; or \; Pb \leftarrow$

$Qa \leftarrow$

has two belief sets $\{Pa, Qa\}$ and $\{Pb, Qa\}$ neither of which contains Qb nor $\neg Qb$. Therefore its answer to a query Qb is *unknown*. The same database viewed as a general disjunctive database answers *no* to Qb. To produce the same answer epistemic specification Π should be extended by the closed world assumption for the predicate Q.

The same phenomena in the context of general logic programs was discussed in [GL90], where it was shown that such an assumption could be expressed by the rule

$$\neg Qx \leftarrow not \; Qx. \tag{4}$$

In the presence of multiple belief sets the situation is more complicated. In this section we will suggest a form of the closed world assumption which differs from the other proposals and will discuss the suitability of this assumption for knowledge representation.

We will start with the following example:

Example 10. [Cha89]. Suppose we are given the following information:

(*) "If a suspect is violent and is a psychopath then the suspect is extremely dangerous. This is not the case if the suspect is not violent or not a psychopath"

which is used in conjunction with a database DB consisting of literals specifying values of the predicates *violent* and *psychopath*. Let us also assume

that DB contains complete positive information about these predicates, i.e. ground atoms with predicate symbols *violent* and *psychopath* are **assumed to be false if there is no reason to suspect that they are true**. This statement can be viewed as an informal description of Reiter's closed world assumption.
The information from (*) can be easily expressed by three rules

1. $dangerous(x) \leftarrow violent(x), psychopath(x)$
2. $\neg dangerous(x) \leftarrow \neg violent(x)$
3. $\neg dangerous(x) \leftarrow \neg psychopath(x)$

Formalization of closed world assumption is somewhat more problematic. The formalization given by formula (4) works nicely for well-defined extended programs but is not suitable in the general case. To see the problem let us apply this idea to our example. Closed world assumptions for predicates *violent* and *psychopath* will look as follows:

4. $\neg violent(x) \leftarrow not\ violent(x)$.
5. $\neg psychopath(x) \leftarrow not\ psychopath(x)$

Suppose that our DB contains the following information:

6. $violent(john) \leftarrow$,
7. $violent(mike) \leftarrow$,
8. $psychopath(mike) \leftarrow$.

It is easy to check that the theory T_1 consisting of clauses (1)–(8) is well-defined and has exactly one belief set A_0:

$$\{violent(john),\ violent(mike),$$

$$psychopath(mike),\ \neg psychopath(john),$$

$$\neg dangerous(john),\ dangerous(mike)\}$$

which properly reflects our intuition. **The situation changes when disjunctive information is allowed in DB**. Consider, for instance, a rule

9. $violent(sam)\ or\ psychopath(sam) \leftarrow$

and a specification T_2 consisting of clauses (1)–(5) and (9). Notice that (9) is not an exclusive disjunction and therefore T_2 does not seem to sanction the conclusion

10. $\neg dangerous(sam)$.

But it is easy to see that T_2 has two belief sets

$$\{violent(sam),\ \neg psychopath(sam),\ \neg dangerous(sam)\}$$

and

$$\{\neg violent(sam),\ psychopath(sam),\ \neg dangerous(sam)\}$$

and therefore implies (10) which seems to be overly optimistic. The problem is apparently caused by an incorrect formalization of the closed world assumption — the fact that *not* $violent(sam)$ is true in one of the belief sets of T_2 does not guarantee that, given T_2, a rational reasoner does not have a reason to believe $violent(sam)$. In the case of T_2 such reason may be given by the existence of a belief set containing $violent(sam)$. This consideration leads to a better representation of the closed world assumption for a predicate P which is provided by the rule

$$\neg P(x) \leftarrow \neg \mathrm{M} P(x) \tag{5}$$

We will later prove that for well-defined programs both formalizations of the closed world assumption coincide.

Let us now consider theory T_3 obtained from T_2 by replacing rules (4) and (5) by

4'. $\neg violent(x) \leftarrow \neg \mathrm{M} violent(x)$.

5'. $\neg psychopath(x) \leftarrow \neg \mathrm{M} psychopath(x)$

The resulting theory has three world views (see Proposition 2 below):

$A_1 = \{\{violent(sam)\},\ \{psychopath(sam)\}\}$,

$A_2 = \{\{violent(sam),\ \neg psychopath(sam),\ \neg dangerous(sam)\}\}$,

$A_3 = \{\{\neg violent(sam),\ psychopath(sam),\ \neg dangerous(sam)\}\}$,

T_3 implies neither (10) nor its negation and therefore answer to the query $dangerous(sam)$ is *unknown*.

The following Proposition describes some basic properties of the operation of adding closed world assumption to the database. We will need the following notation and definitions. Let $A = \{A_i\}$ be a collection of sets of ground literals. By S_A we will denote the union of all literals from the elements of the set A, (i.e. $S_A = \cup\{A_i\}$). Finally, $\overline{\mathcal{A}} = \{\neg Pt : Pt \in Lit \setminus S_A\}$. We will say that $A = \{A_i\}$ *covers* a set B of ground literals if for every literal $l \in B$ there is i such that $l \in A_i$.

Let Π be a general disjunctive database and $A = \{A_i\}$ be a collection of its belief sets. We will say that A is *saturated* if it contains every belief set of Π covered by A.

For instance, if $\Pi = \{Pa \; or \; Pb \leftarrow, \; Pc \; or \; Pd \leftarrow\}$ then the set $A = \{\{Pa, Pc\}\}$ is saturated while the set $A_0 = \{\{Pa, Pc\}, \{Pb, Pd\}\}$ is not (since it does not contain, say, $\{Pa, Pd\}$ which is a belief set of Π and is covered by A_0.) Obviously, the collection of all belief sets of Π is saturated.

Π^* will stand for the epistemic specification obtained from Π by adding to it the set C of all the rules of the form

$$\neg Px \leftarrow \neg \mathrm{M} Px$$

where P is a predicate symbol from the language of Π.

Proposition 2. Let Π be a general disjunctive database. Then A^* is a consistent world view of Π^* iff there is a saturated collection A of belief sets of Π such that $A^* = \{V : \exists W \; (W \in A \; \& \; V = W \cup \overline{\mathcal{A}})\}$.

To prove Proposition 2 we will need the following Lemmas.

Let Π be an extended disjunctive database. For any predicate P occurring in Π, let P' be a new predicate of the same arity. Π^+ will stand for the general disjunctive database obtained from Π by replacing all occurrences of negative literals $\neg P(\ldots)$ by $P'(\ldots)$. For any $W \subset Lit$, W^+ is obtained from W by replacing negative literals $\neg P(\ldots)$ from W by $P'(\ldots)$.

Lemma 1. A consistent set $W \subset Lit$ is a belief set of an extended disjunctive database Π iff W^+ is a belief set of Π^+.

Proof. Proof of this Lemma is similar to the proof of Proposition 2 in [GL91] and will be left to the reader.

Let W and $\mathcal{U}$ be sets of ground literals and A be a collection of sets of such literals. Then $W \cap \mathcal{U}$ will be denoted by $W^{\mathcal{U}}$ and a set $\{W^{\mathcal{U}} : W \in A\}$ will be denoted by $A^{\mathcal{U}}$.

Lemma 2. Let Π^1 and Π^2 be epistemic specifications whose rules do not contain quantifiers and let α and β be the sets of all ground literals in the languages of Π^1 and Π^2 respectively. If α and β are disjoint then a collection A of sets of ground literals is a world view of $\Pi = \Pi^1 \cup \Pi^2$ iff A^α is a world view of Π^1 and A^β is a world view of Π^2.

Proof. Let us first prove that if Π is an extended disjunctive database then

(*) W is a belief set of Π iff W^α and W^β are belief sets of Π^1 and Π^2 respectively.

(a) If Π does not contain negation as failure *not* the statement is obviously true.

(b) Suppose now that $\Pi = \Pi^1 \cup \Pi^2$ is an arbitrary extended disjunctive database. Recall that W is a belief set of Π iff it is a belief set of the corresponding reduct Π_W. Since α and β are disjoint,

$\Pi_W = \Pi^1_{W^\alpha} \cup \Pi^2_{W^\beta}$.

Since Π_W does not contain *not*, this, together with (a), implies (*).

(c) Now let Π be an arbitrary epistemic specification. A is a world view of Π iff A is equal to the collection of all belief sets of an extended disjunctive database Π_A. Obviously,

$\Pi_A = \Pi^1_{A^\alpha} \cup \Pi^2_{A^\beta}$.

which, together with (b), implies the conclusion of the Lemma.

Lemma 3. If an extended disjunctive database Π is consistent then Π has no inconsistent belief set.

Proof. Let us assume that Lit is a belief set of Π and partition the database Π into two parts: Π_0 containing all rules of Π without occurrences of negation as failure *not* and Π_1 containing the rest of the rules. By definition, Lit is a belief set of Π iff it is a belief set of Π_{Lit}. It is easy to see that $\Pi_{Lit} = \Pi_0$ and hence Lit must be a belief set of Π_0. To see that it is impossible notice that since Π is consistent it has a consistent belief set $W \subset Lit$ which is obviously

closed under the rules of Π_0 and smaller than Lit. Hence Lit can not be a belief set of Π which contradicts our assumption.

Proof of Proposition 2.

Proof. (a) Let A be a saturated collection of belief sets of Π. For every $W \in A$ we will define the set $W^* = W \cup \overline{\mathcal{A}}$ and show that $A^* = \{W^* : W \in A\}$ is a world view of Π^*.

Let us first notice that

(1) $(\Pi^*)_{A^*} = \Pi_{A^*} \cup C_{A^*}$

Since Π does not contain $\neg$, definition of the reduct implies

(2) $\Pi_{A^*} \cup C_{A^*} = \Pi_A \cup C_A$

and, by definitions of the reduct and of the sets C and $\overline{\mathcal{A}}$

(3) $\Pi_A \cup C_A = \Pi \cup \overline{\mathcal{A}}$

This implies that A^* is a world view of Π^* iff A^* is the collection of all belief sets of $\Pi \cup \overline{\mathcal{A}}$.

First let us demonstrate that every $W^* \in A^*$ is a belief set of $\Pi \cup \overline{\mathcal{A}}$. It is easy to see that W^* is consistent and therefore, by Lemma 1, W^* is a belief set of $\Pi \cup \overline{\mathcal{A}}$ iff $V = (W^*)^+$ is a belief set of a specification $(\Pi \cup \overline{\mathcal{A}})^+$. Obviously,

(4) $(\Pi \cup \overline{\mathcal{A}})^+ = \Pi \cup \overline{\mathcal{A}}^+$.

Since languages of Π and $\overline{\mathcal{A}}^+$ are disjoint, by Lemma 2 and the definition of V we have that V is a belief set of $\Pi \cup \overline{\mathcal{A}}^+$, and therefore W^* is a belief set of $\Pi \cup \overline{\mathcal{A}}$. Observe also that, since W^* is consistent, the above argument implies consistency of $\Pi \cup \overline{\mathcal{A}}$.

Now let us assume that V is a belief set of $\Pi \cup \overline{\mathcal{A}}$ and show that $V \in A^*$. From the consistency of $\Pi \cup \overline{\mathcal{A}}$ and Lemma 3 we conclude that V is consistent. By virtue of Lemma 1 V^+ is a belief set of $\Pi \cup \overline{\mathcal{A}}^+$. By Lemma 2 we have that $V = W \cup \overline{\mathcal{A}}$ where W is a belief set of Π. Suppose that $W \notin A$. Since A is saturated this implies that W is not covered by A, i.e. there is some ground atom $p(t) \in W$ such that $\neg p(t) \in \overline{\mathcal{A}}$. This implies that V is inconsistent which contradicts our assumptions. Therefore, $V \in A^*$ which completes part (a) of the proof.

(b) Assume that A^* is a consistent world view of Π^* and show that there is a saturated collection A of belief sets of Π such that $A^* = \{W^* : \exists\ W\ (W \in A\ \&\ W^* = W \cup \overline{\mathcal{A}})\}$.

Let $A = \{W : \exists V(V \in A^* \& W = V \cap Atoms)\}$. Using an argument similar to the one above we can demonstrate that

(5) $W^* \in A^*$ iff W^* is a belief set of $\Pi \cup \overline{\mathcal{A}}$.

First let us consider $W^* \in A^*$. Since A^* is consistent, so is W^* and therefore, by Lemma 2, there is W such that W is a belief set of Π and $W^* = W \cup \overline{\mathcal{A}}$. By construction of A we immediately conclude that $W \in A$.

Now consider $W \in A$ and $W^* = W \cup \overline{\mathcal{A}}$. We will show that $W^* \in A^*$. Observe, that A^* is a world view of $\Pi \cup \overline{\mathcal{A}}$ and hence, by construction of A and Lemma 2 we have that any $W \in A$ is a belief set of Π. This implies that W^* is a belief set of $\Pi \cup \overline{\mathcal{A}}$, and, by (5), $W^* \in A^*$.

Finally we will show that A is saturated. Let W_0 be a belief set of Π covered by A. From definition of $\overline{\mathcal{A}}$ we have that $(W_0)^* = W_0 \cup \overline{\mathcal{A}}$ is consistent and, by Lemma 2 $(W_0)^*$ is a belief set of $\Pi \cup \overline{\mathcal{A}}$. By (5) we have that $(W_0)^* \in A^*$ and hence, by the construction of A, $W \in A$.

Corollary 1. Let Π be a general disjunctive database, B be a collection of all belief sets of Π and let

$B^* = \{W^* : \exists\ W\ (W \in B\ \&\ W^* = W \cup \overline{\mathcal{B}})\}$.

Then for any literal l, $\Pi^* \models l$ iff l is true in B^*.

Proof. Follows immediately from Proposition 2 and the fact that B is saturated.

Corollary 2. Let Π be a general disjunctive database. For any ground atom $p(t)$, $\Pi^* \models p(t)$ iff $\Pi \models p(t)$.

Proof. Follows immediately from Corollary 1.

Proposition 2 implies that if a general disjunctive database Π is consistent then so is Π^* and that no new positive information can be obtained by expanding Π by the closed world assumption. This is no longer true if Π contains classical negation $\neg$. Consider, for instance, a database

$\Pi = \{Q \leftarrow \neg P\}$.

Obviously, Π does not entail Q while Π^* does. To get an example of a consistent database whose closure is inconsistent consider

$\Pi = \{Q \leftarrow \neg P\ ,\ \neg Q \leftarrow\}$.

It will be interesting to find broader classes of epistemic specifications satisfying conclusion of Proposition 2.

The following Corollary demonstrates that for well-defined general logic programs our formalization of closed world assumption coincides with the one from [GL91]. More precisely, we have

Corollary 3. Let Π be a consistent general logic program with a unique stable model. Let Π^* be as in Proposition 2 and Π^{**} be an extended logic program obtained from Π by adding to it the rule $\neg Px \leftarrow not\ Px$ for every predicate symbol P from the language of Π. Then for every ground query Q in the language of Π Q is $true$ $(false)$ in Π iff Q is $true$ $(false)$ in Π^{**} is $true$ $(false)$ in Π^*.

Proof. The first equivalence follows from Proposition 4 of [GL91], while the second is an immediate consequence of Propositions 1 and 2 above.

In the remainder of this section we will briefly discuss the relationship between our form of closed world assumption and other forms incorporated in the known semantics of disjunctive databases. First we will demonstrate that, in the presence of the closed world assumption, epistemic semantics for positive disjunctive databases coincides with the semantics from [Min82].

We will abuse the notation and use Π to denote a positive disjunctive database as well as a first–order theory obtained from it by replacing every rule

$$A_1\ or\ \ldots\ or\ A_n \leftarrow B_1, \ldots, B_m \tag{6}$$

by a formula

$$B_1 \& \ldots \& B_m \supset A_1 \vee \ldots \vee A_n \tag{7}$$

Recall that Minker's generalized closed world assumption is defined as follows. (We will use the terminology from [GP86].) A disjunction D of ground atoms is called *essential* w.r.t. theory Π if $\Pi \models D$ and no subdisjunction of D is entailed by Π. A ground atom is called *free for negation* in Π if it does

not belong to any clause essential in Π. Let $\overline{\Pi}$ be a set of negations of all ground atoms free for negation in Π. Then

$GCWA(\Pi) = \Pi \cup \overline{\Pi}$.

Let us denote the set of all Herbrand models of $GCWA$ by $\mathcal{M}$. (We identify a Herbrand model with the set of ground atoms true in this model). For simplicity, we will limit ourself to positive ground queries of the form $P(c_1) \vee \ldots \vee P(c_n)$ and their negations. To simplify the notation we will use the same letter Q to denote its epistemic counterpart $P(c_1)\ or\ \ldots\ or P(c_n)$.

According to [Min82], for every query Q, the $GCWA$ answer to Q is *yes* if Q is true in all models from $\mathcal{M}$, (which will be denoted by $\mathcal{M} \models Q$), *no* if $\neg Q$ is true in all such models, and *unknown* otherwise. The *closed world answer* to an (epistemic) query Q by a database Π is the answer to Q by Π^*.

Proposition 3. For any positive disjunctive database Π and any query Q, the $GCWA$ answer to Q coincides with the closed world answer to Q.

Proof. First let us recall that, as was proved for a finite theory Π in [Min82] and for an arbitrary Π in [GP86], $\overline{\Pi}$ is equal to the set of negations of all ground atoms not belonging to any minimal Herbrand model of Π. Obviously, M is a minimal model of Π iff M is a belief set of Π and hence, by Proposition 2 a set $B = \{W : W = M \cup \overline{\Pi}\}$ is a world view of Π^*. Recall, that by Corollary 1, for any ground atom $p(\ldots)$, $\Pi^* \models p(\ldots)$ iff $p(\ldots)$ belongs to all belief sets of B. Consider two cases.
(a) Q is a disjunction of atoms. Then it is easy to see that $\mathcal{M} \models Q$ iff Q is true in all minimal Herbrand models of Π iff Q is true in all belief sets of B iff $\Pi^* \models Q$.
(b) Q is a conjunction of negative literals. Then the conclusion of the proposition follows from the fact that for any atom $p(\ldots)$, $\mathcal{M} \models \neg p(\ldots)$ iff $\neg p(\ldots) \in \overline{\Pi}$.

The following examples demonstrate differences between our semantics and several others.

(a) Consider the semantics based on the extended closed world assumption (ECWA) [GPP86, YH85], and its extensions such as perfect models semantics [Prz88], stationary semantics [Prz90], etc.

Let $\Pi = \{Pa\ or Pb \leftarrow\}$. The ECWA answer to a query $Pa\&Pb$ is *no*, while the closed world answer to $Pa\&Pb$ is *unknown*.

(b) Consider possible world types of semantics [Cha89, RT88, C.89] etc.

Let $\Pi = \{Pa \; or \; Pb \leftarrow, \quad Pa \leftarrow\}$.

It is easy to see that the closed world answer to a query Pb is *no*, while the answer based on possible world semantics is *unknown*.

3.3 Unique Name Assumption

The unique name assumption [Rei78] is normally used in the settings when one can assume that all the relevant information about the equality of individuals has been specified. In this case **all pairs of individuals not specified as identical are assumed to be different**. To express this assumption we will expand the language $\mathcal{L}_T$ by the binary predicate symbol E which stands for *equality*.

The following rules can be viewed as the definition of E:

$E(x, x) \leftarrow$

$E(x, y) \leftarrow E(y, x)$

$E(x, y) \leftarrow E(x, z), E(z, y)$

$F(y) \leftarrow E(x, y), F(x)$

for every objective literal F.

$\neg E(x, y) \leftarrow \neg \mathrm{M} E(x, y)$

These rules will be called *predefined*. From now on, we will only consider specifications containing these rules.

Example 10. Unique name assumption.

Suppose that our language $\mathcal{L}_T$ contains the list of names such as $mike$, $john$, $mary$, etc., and assume that the specification T_1 includes the following complete list of professors in a computer science department:

1. $P(mike, cs) \leftarrow$

2. $P(john, cs) \leftarrow$

To express the completeness of the list we will use the closed world assumption

3. $\neg P(x, y) \leftarrow \neg \mathrm{M} P(x, y)$.

The world view of the resulting theory consists of belief sets containing

$\{P(mike, cs), P(john, cs)\}$

and the negations of all other atoms of the form

$P(c_1, c_2)$

where c_1, c_2 are other pairs of constants from $\mathcal{L}_T$.

Let us now assume that Mike also goes by another name, say, Misha. This information can be coded in our system as

4. $E(mike, misha) \leftarrow$.

The belief sets of the new specification (1) – (4) can be obtained from the old belief sets by replacing $\neg E(misha, mike)$ and $\neg P(misha, cs)$ by $E(misha, mike)$ and $P(misha, cs)$ respectively.

3.4 Normative Statements

In this section we will discuss representation of statements of the form

"P**'s are normally (typically, as a rule, etc.)** Q**'s** "

(called normative statements). We suggest to code such statements by the rules of the form

$Q(x) \leftarrow P(x), not\ AB(q, p, x), not\ \neg Q(x)$

where AB is an abnormality predicate from [McC80] and q and p are object constants corresponding to predicate constants Q and P. This coding can be viewed as a combination of the representation of normative statements in circumscription with the method used in non-monotonic modal logics [McC80, MD80]. It was also advocated in [PCA91].

For illustration, let us consider database T_1 from Example 10 and expand it by the following information:

"*As a rule, professors in the computer science department have vax accounts. This rule is not applicable to Mike. He may or may not have an account.*"

As suggested above, the first statement will be represented by the rule:

5. $A(x, vax) \leftarrow P(x, cs), not\ AB(a, p, x), not\ \neg A(x, vax)$

Where A(x,y) stands for "x has an account on y", while the second statement is translated as

6. $AB(a, p, mike) \leftarrow$

It is easy to see that the resulting theory entails $A(john, vax)$ but stays undecided about Mike (and Misha). The rule (6) allows us to block the application of the rule (5) without refuting its conclusion. Suppose now we've learned that "*there is another exceptional professor (say Greg) who does not have a vax account.*" To reflect this knowledge we update the database by the rules

7. $P(greg, cs) \leftarrow$

8. $\neg A(greg, vax) \leftarrow$

Notice, that in this case, the application of the normative rule is blocked by the defeasible part of the premise.

The presence of normative statements can substantially complicate the process of translation from English to the language of epistemic specifications (as well as to other known logical formalisms). The following demonstrates some of the difficulties: Suppose we've learned that

"*every computer science professor has one of the vax or ibm accounts, but not both.*"

In the absence of any other information about computer accounts this can be represented, say, by the rules:

9. $A(x, vax)\ or A(x, ibm) \leftarrow P(x, cs)$

10. $\neg A(x, ibm) \leftarrow A(x, vax)$

11. $\neg A(x, vax) \leftarrow A(x, ibm)$

In conjunction with the rules from Example 10 this formalization produces the intuitive results and is also sufficiently simple. The situation changes however when we add the rules (5) - (8). We expect the resulting theory T_3 to conclude, among other things, that John has a vax account. This is, however, not the case, since there are two belief sets in the world view of T_3, one containing A(john,vax) and another containing A(john,ibm). The problem occurs because of the two contrary rules (5) and (11) which can both be applied to the same professor x, and no priority is given to the rule

(5). The correct solution requires a finer analysis of the situation. First we should notice that *the rule (5) should be used whenever possible* and that *the new information is only applicable to the professors which are exceptions to (5)*. Two types of exceptions are possible: firstly, we may know that a professor x does not have a vax account. In this case we should instruct our database to believe that x has an ibm account. This is easily done by adding to it the rule:

12. $A(x, ibm) \leftarrow \neg A(x, vax), P(x, cs)$

Now the predicate A is undefined only for the professors known to be abnormal in some respects. For such professors we have no reason to prefer neither ibm or vax and this lack of preference is reflected by the rule

13. $A(x, vax)\, or\, A(x, ibm) \leftarrow P(x, cs), ab(a, p, x)$

The next two rules insure that no computer science professor has both accounts

14. $\neg A(x, vax) \leftarrow P(x, cs), A(x, ibm)$

15. $\neg A(x, ibm) \leftarrow P(x, cs), A(x, vax)$

It is easy to see that now the database from Example 10 together with the rules (5) – (8) and (12) – (15) entails that John has a Vax account, Greg has one for the IBM and Mike (Misha) is still undecided (He has IBM's in one of the belief sets and VAX's in the other one).

It is too early to say if there is a uniform, formalizable strategy for translating disjunctive information of the type discussed above into epistemic specifications. It is clear, however, that developing methodology of such translation (similar to the development of programming methodologies in more conventional languages) is a necessary step in answering this question.

3.5 Extended Queries

In the previous examples epistemic specifications were used to answer simple queries about their knowledge of the world. The next example demonstrates how they can answer more complicated queries. The following story was discussed (in a somewhat different context) in [BLM91].

Example 11. Assume that we are preparing for a camping trip and we are deciding what equipment we are going to take on the trip. We would like

to bring with us a stove and a blanket. However, we have learned from our previous trip that we have some restrictions. We know that we can take the stove only if we do not bring the tent and we can bring the blanket only if we do not bring the mat. We also know that to protect the blanket from the stove we need a plastic cover to cover the blanket. We need shelter from the rain, so we have to bring either the tent or the mat. The problem is to check if these restrictions allow us to take both the stove and the blanket.

It is easy to see that the restrictions can be represented by the following epistemic specification T_4:

1. $\neg tent \leftarrow stove$
2. $\neg mat \leftarrow blanket$
3. $cover \leftarrow stove,\ blanket$
4. $tent\ or\ mat \leftarrow$

Now let us consider the rules

5. $stove \leftarrow$
6. $blanket \leftarrow$

Obviously, taking both the stove and the blanket satisfies our restrictions iff database T_4 expanded by the last two rules is consistent. It is easy to see that it is not the case, and therefore, we can not fully satisfy our wish. We can however take with us the tent and the blanket, the stove and the mat, etc. Here rules (5) and (6) are used as a more sophisticated query which significantly uses the notion of inconsistent database. (We could, of course ask the database to give us a maximal subset of a given set of literals consistent with our restrictions.)

3.6 Integrity Constraints

We will finish our discussion of applicability of strong introspection to formalization of commonsense reasoning by an example demonstrating the utility of strong introspection for expressing integrity constraints.

Example 12. Let us assume that we are given the specification for a departmental database T:

(a) T should contain lists of professors, courses and teaching assignments for a CS department. Let us first assume that the department consists of pro-

fessors named Sam and John and offers two courses: Pascal and Assembler, taught by Sam and John respectively.

(b) The above lists contain all the relevant positive information about the department known to us at a time.

(c) T must satisfy the following integrity constraint: Pascal is taught by at least one professor.

Part (a) of the specification is formalized as follows:

1. $prof(sam) \,\&\, prof(john) \leftarrow$
2. $class(pascal) \,\&\, class(assembler) \leftarrow$
3. $teach(sam, pascal) \,\&\, teach(john, assembler) \leftarrow$

Part (b) of the specification can be viewed as the Closed World Assumption and represented as

4. $\neg P(x) \leftarrow \neg \mathrm{M} P(x)$

for every predicate symbol P.

Formalization of part (c) seems to be the less obvious task. The main difficulty is related to the lack of universally accepted interpretation of the meaning and the role of integrity constraints in knowledge representation. In this paper we will adopt the view on integrity constraints recently suggested by Reiter in [Rei90]. According to Reiter an integrity constraint IC is a statement about the content of the knowledge base T (as opposed to IC being a statement about the world). T satisfies IC iff the answer to IC when viewed as a query to T is *yes*. A simple analysis of clause (c) from this standpoint shows that (c) can be interpreted in two different ways:

$IC_1 : \mathrm{K}\ \exists p(prof(p) \,\&\, teach(p, pascal))$

or

$IC_2 : \exists p\ \mathrm{K}\ (prof(p) \,\&\, teach(p, pascal))$

The first one says that the database knows that Pascal is taught by some professor (whose name can be unknown to the database), while the second one means that there is a person known to the database to be a professor teaching Pascal. It is easy to see that T_1 consisting of rules (1)–(3) satisfies both integrity constraints.

If however, we consider T_2 obtained from T_1 by replacing rule 2 by

2'. $teach(sam, pascal)$ *or* $teach(john, pascal)$

the situation changes.

It is easy to check that T_2 satisfies IC_1 but not IC_2. This is, of course, the intended result since this time the database does not know what professor will teach Pascal but knows that Pascal will be taught.

Remark. Even though the approach to formalization of integrity constraint suggested in this paper is similar to the one of Reiter there are some important differences: Reiter views a knowledge base as a first-order theory and a query as a statement of Levesque's modal logic (called KFOPCE). In our case knowledge base and queries are both epistemic formulae while the underlying logic is nonmonotonic.

4 Appendix

In this section we will briefly discuss the relationship between epistemic specifications and some other general purpose nonmonotonic formalisms such as autoepistemic logics [MD80], [Moo85], [MT90], and default logic of R. Reiter [Rei80] and its extension [GLPT91]. By now we have a reasonably good understanding of the relationship between these formalisms and special classes of epistemic specifications. For the equivalence results about various subclasses see for instance [Gel87], [BF87], [GLPT91]. It is more difficult to use these formalisms to model modal operators of epistemic specifications. Autoepistemic logics, which seem to be natural candidates for such modeling, apparently do not work. There are several important differences between them and epistemic specifications two of which are probably most important. First, the language of autoepistemic logic is a simple extension of the language of propositional calculus with its standard set of classical connectives $\&, \vee, \supset, \neg$ while epistemic specifications use different set of connectives.

Second, even though epistemic statement Kp as well as its autoepistemic counterpart $\mathbf{L}p$ are both translated in English as "know (or believe) p" they refer to different knowing abilities of a reasoning agent. The former has strong (or global) power of introspection and concludes Kp by simultaneously looking at a whole collection of possible belief sets corresponding to his set of premises T while the latter is only capable of looking at one of such sets at a time. His conclusion about the truth of $\mathbf{L}p$ is local and based on the

set of statements he currently beliefs in. This makes it difficult to express epistemic closed world assumption discussed in this paper in autoepistemic logic. Similar problem arises when default logic is used for the same purpose. A more detailed discussion of this (as well as some ideas about computing world views of epistemic specifications) can be found in [GP91].

Acknowledgments

I would like to thank Vladimir Lifschitz, Halina Przymusinska, Marek Suchenek, Bonnie Traylor and Thomas Woo for suggestions on a draft of this paper. Special thanks to Teodor Przymusinski whose comments helped to discover an error in the original draft. This work was supported in part by NSF grants IRI–9103112 and CDA–9015006.

References

[BF87] Nicole Bidoit and Christine Froidevaux. Minimalism subsumes default logic and circumscription. In *Proc. of LICS-87*, pages 89–97, 1987.

[BLM91] Chitta Baral, Jorge Lobo, and Jack Minker. Wf3: A semantics for negation in normal disjunctive logic programs. In *Proc. of International Symposium on Methodologies of Intelligent Systems*, 1991.

[C.89] Sakama C. Possible model semantics for disjunctive databases. In *Proc. of the first international conference on deductive and object oriented databases*, pages 1055–1060, 1989.

[Cha89] Edward Chan. A possible world semantics for non–horn databases. Technical Report CS–89–47, University of Waterloo, 1989.

[Cla78] Keith Clark. Negation as failure. In Herve Gallaire and Jack Minker, editors, *Logic and Data Bases*, pages 293–322. Plenum Press, New York, 1978.

[Gel87] Michael Gelfond. On stratified autoepistemic theories. In *Proc. AAAI-87*, pages 207–211, 1987.

[Gel91] Michael Gelfond. Strong introspection. In *Proc. AAAI-91*, pages 386–391, 1991.

[GL88] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In Robert Kowalski and Kenneth Bowen, editors, *Logic Programming: Proc. of the Fifth Int'l Conf. and Symp.*, pages 1070–1080, 1988.

[GL90] Michael Gelfond and Vladimir Lifschitz. Logic programs with classical negation. In David Warren and Peter Szeredi, editors, *Logic Programming: Proc. of the Seventh Int'l Conf.*, pages 579–597, 1990.

[GL91] Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, pages 365–387, 1991.

[GLPT91] Michael Gelfond, Vladimir Lifschitz, Halina Przymusińska, and Miroslaw Truszczyński. Disjunctive defaults. In James Allen, Richard Fikes, and Erik Sandewall, editors, *Principles of Knowledge Representation and Reasoning: Proc. of the Second Int'l Conf.*, pages 230–237, 1991.

[GP86] Michael Gelfond and Halina Przymusinska. Negation as failure: Careful closure procedure. *Artificial Intelligence*, 30(3):273–287, 1986.

[GP91] Michael Gelfond and Halina Przymusińska. Definitions in epistemic specifications. In Anil Nerod, Victor Marek, and Subramanian V. S., editors, *Logic Programming and Non–monotonic Reasoning: Proc. of the First Int'l Workshop*, pages 245–259, 1991.

[GPP86] Michael Gelfond, Halina Przymusinska, and Teodor Przymusinski. The extended closed world assumption and its relation to parallel circumscription. In *Proc. of the fifth ACM Symposium on Principles of Database Systems*, pages 133–139, 1986.

[KS90] Robert Kowalski and Fariba Sadri. Logic programs with exceptions. In David Warren and Peter Szeredi, editors, *Logic Programming: Proc. of the Seventh Int'l Conf.*, pages 598–613, 1990.

[Lev86] Hector Levesque. Making believers out of computers. *Artificial Intelligence*, 30:81 – 108, 1986.

[Lev90] Hector Levesque. All I know: a study in autoepistemic logic. *Artificial Intelligence*, 42(2,3):263–310, 1990.

[LMR92] Jorge Lobo, Jack Minker, and Arcot Rajasekar. *Foundations of disjunctive logic programming*. The MIT Press, 1992.

[LT84] John Lloyd and Rodney Topor. Making prolog more expressive. *Journal of Logic Programming*, 3:225–240, 1984.

[McC80] John McCarthy. Circumscription—a form of non-monotonic reasoning. *Artificial Intelligence*, 13(1, 2):27–39,171–172, 1980.

[MD80] Drew McDermott and Jon Doyle. Nonmonotonic logic I. *Artificial Intelligence*, 13(1,2):41–72, 1980.

[Min82] Jack Minker. On indefinite data bases and the closed world assumption. In *Proc. of CADE-82*, pages 292–308, 1982.

[Moo85] Robert Moore. Semantical considerations on nonmonotonic logic. *Artificial Intelligence*, 25(1):75–94, 1985.

[MT90] Wiktor Marek and Miroslaw Truszczyński. Modal logic for default reasoning. To appear, 1990.

[PCA91] Luis Pereira, Luis Caires, and Jose Alferes. Hypothetical reasoning with well founded semantics. In *Proc. of the 3rd Scandinavian Conference on AI*, 1991.

[PP90] Halina Przymusinska and Teodor Przymusinski. Semantic issues in deductive databases and logic programs. In R Manerji, editor, *Formal Techniques in Artificial Intelligence*, pages 321 – 367. North Holland, Amsterdam, 1990.

[Prz88] Teodor Przymusinski. On the declarative semantics of deductive databases and logic programs. In Jack Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 193–216. Morgan Kaufmann, San Mateo, CA, 1988.

[Prz90] Teodor Przymusinski. Extended stable semantics for normal and disjunctive programs. In David Warren and Peter Szeredi, editors, *Logic Programming: Proc. of the Seventh Int'l Conf.*, pages 459–477, 1990.

[PW89] David Pearce and Gerd Wagner. Reasoning with negative information 1 – strong negation in logic programming. Technical report, Gruppe fur Logic, Wissentheorie and Information, Freie Universitat Berlin, 1989.

[Rei78] Raymond Reiter. On closed world data bases. In Herve Gallaire and Jack Minker, editors, *Logic and Data Bases*, pages 119–140. Plenum Press, New York, 1978.

[Rei80] Raymond Reiter. A logic for default reasoning. *Artificial Intelligence*, 13(1,2):81–132, 1980.

[Rei90] Raymond Reiter. On asking what a database knows. In John Lloyd, editor, *Computational Logic: Symposium Proceedings*, pages 96–113. Springer, 1990.

[RT88] Kenneth Ross and Rodney Topor. Inferring negative information from disjunctive databases. *Journal of Automated Reasoning*, 4(4):397–424, 1988.

[Wag91] Gerd Wagner. Database needs two kinds of negation. In *Proc. of MFDBS–91, (Lecture Notes in Computer Science, 495)*, 1991.

[YH85] L. Yahya and A Henschen. Deduction in non–horn databases. *Journal of Automated Reasoning*, 1(2):141–160, 1985.